Proceedings of CONF-SPML 2026 Symposium: The 2nd Neural Computing and Applications Workshop 2025
DOI: 10.54254/2755-2721/2026.TJ31027

Data Privacy Risks of Large Language Models: A Comprehensive Review

Junkai Peng

Tongji University, Shanghai, China
2353931@tongji.edu.cn

Abstract. Large Language Models (LLMs) achieve strong performance across domains, yet persistent privacy risks remain. This paper characterizes privacy for LLMs as control over the collection, processing, sharing, and retention of sensitive information. Risks are grouped into data privacy and model privacy, then mapped to concrete attack surfaces. Data risks include training data leakage and membership inference. Model risks include model inversion, chain-of-thought leakage, Mixture-of-Experts routing leakage, and side-channel observation. Mitigation is organized at three technical levels and one governance layer. Data-level measures include cleaning, de-identification, and synthetic data. Model-level measures include differential privacy training, federated learning, chain-of-thought control, and backdoor defense. System-level measures include authenticated encryption, intrusion detection, traffic shaping, man-in-the-middle defense, and trusted hardware. Regulatory frameworks and standards offer reference points for deployment. Open problems include private inference for long contexts, quantitative privacy budgets in production telemetry, robust defenses for reasoning leakage, and integration with sparse MoE architectures. This perspective clarifies evaluation criteria and informs deployment choices under constraints.

Keywords: Large Language Models (LLMs), Mixture of Experts (MoE), Data Privacy Risks, Privacy Protection.

1. Introduction

Large Language Models (LLMs) have become central to artificial intelligence. The rise of LLMs also brings serious privacy concerns. Training data often contain personal or corporate information. The models may memorize and reproduce such data during text generation. This behavior creates a risk of privacy leakage. Privacy is a basic human right. In China, it is protected by the Data Security Law and the Personal Information Protection Law. These laws require strict control over data collection, storage, and use. However, research on privacy risks in LLMs remains limited. Most existing work focuses on model performance, robustness, or security. Few studies explore privacy directly. This lack of attention leaves hidden vulnerabilities unaddressed. Privacy risks exist throughout the entire lifecycle of LLMs [1]. They appear in data collection, model training, and inference. They may also occur during user interaction or network communication. Sensitive data can be exposed through model outputs, parameters, or system integration.

DOI: 10.54254/2755-2721/2026.TJ31027

2. LLM architecture and its vulnerabilities

LLMs have achieved strong performance through architectural innovation. Early models used dense transformer layers with fixed parameter paths. Later models adopted sparse structures and distributed computation. Modern architectures use token routing, attention, and expert selection. Each method improves capability but adds potential leakage points. Information moves across devices, memory blocks, and servers. Every transfer may expose hidden data. The Mixture of Experts (MoE) architecture shows this issue clearly [2]. It improves speed and specialization through distributed expert routing.

2.1. MoE architecture

The MoE architecture divides computation among multiple experts. Each expert processes a specific part of the input. A gating network selects the most relevant experts for each token. The model then combines their outputs to form the final representation. The overall workflow of MoE-based LLMs is illustrated in Figure 1. The figure shows how input tokens move through embedding and attention layers, enter the gating network, and are routed to the selected experts.

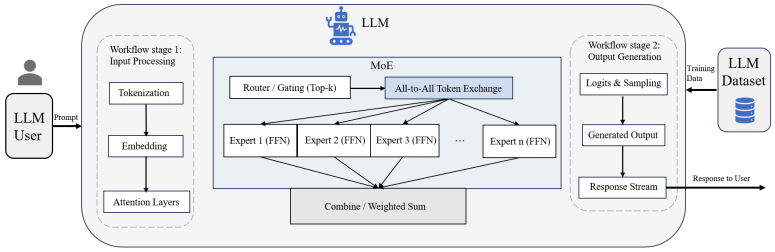

Figure 1. Workflow of MoE architecture in LLMs

The workflow begins with the user prompt. Text input passes through tokenization, embedding, and attention layers. These layers capture semantic and contextual relationships between tokens. The processed representations then enter the MoE layer. A router, also called a gating network, assigns importance scores to all experts. It activates only the top-k experts for each token. Each expert is usually implemented as a Feed-Forward Network (FFN). After computation, outputs are aggregated by the weighted-sum operation above.

2.2. External vulnerabilities

2.2.1. Parameter scaling

LLMs contain billions of parameters that increase their ability to represent language and context. Parameter scaling enhances fitting capacity and semantic coverage. It follows known scaling laws in which performance improves with larger data and more computation. At the billion-to-trillion parameter level, models often reproduce training fragments verbatim [3]. This effect is commonly termed memorization. It creates a direct path for data leakage.

2.2.2. Massive training data

Training LLMs depends on large and diverse datasets. These datasets include public web data, licensed software code, and open-source collections. According to current regulations, the use of

DOI: 10.54254/2755-2721/2026.TJ31027

training data must follow the Interim Measures for the Management of Generative Artificial Intelligence Services [4] and the Technical Specification for Data Security in Pre-training and Optimization Training of Generative Artificial Intelligence [5]. Unauthorized data collection is still reported. Sensitive information such as registration codes and software serial numbers is often improperly included in training datasets.

2.2.3. Fine-tuning

Fine-tuning adapts a general LLM to specific tasks or domains. It uses additional datasets to align the model with domain-specific language and knowledge. This method improves accuracy and relevance for targeted applications. Fine-tuning datasets often contain sensitive or confidential information [6]. During this process, the model can encode such information into its parameters. Later, the model may reproduce it during response generation. This behavior exposes private data to users or attackers. Domain-specific LLMs provide strong performance but increase privacy risk.

2.2.4. Generative capabilities

LLMs are generative models. They produce natural language text based on input prompts. This ability comes from autoregressive training on large datasets. During training, the model learns language patterns by predicting the next token. The generative ability enables LLMs to create fluent and context-aware responses. They can synthesize information, restate knowledge, and produce text similar to human writing.

2.2.5. Communication mechanism

Some LLM deployments operate within intranet environments. In such settings, user–model data may still be transmitted to external servers. If this communication uses unencrypted protocols, attackers can intercept sensitive information. For cloud-based LLMs, stored interaction data may face additional risks. Weak cloud security or poor access control can lead to data theft. Attackers can also exploit prompt injection to force the model to reveal private content from its training corpus. Using insecure network channels, such as HTTP instead of HTTPS, increases the risk of interception. Even encrypted channels require secure configuration.

2.2.6. Hardware infrastructure

Many private LLM servers face hardware and network security issues. Some systems remain directly exposed to the public Internet. They often lack password protection or basic firewall settings. Attackers can easily access such servers and steal stored data. Smaller service providers usually deploy open-source frameworks without strengthening their default configurations. These weak points allow unauthorized users to exploit GPU resources or disrupt model services. In some cases, attackers use these interfaces to overload hardware and cause service interruption. Hardware-level threats are also critical. Core computing components such as GPUs or NPUs may contain hidden vulnerabilities or backdoors.

3. Key privacy risks in LLMs

Privacy is the right of an individual or organization to control how information about them is collected, processed, and shared. In LLMs, this definition requires extension. The models learn from

DOI: 10.54254/2755-2721/2026.TJ31027

large datasets and encode complex patterns in internal representations. They may thus expose sensitive content in ways beyond traditional data flows. The first dimension is data privacy. It covers the protection of sensitive or identifiable information in training and fine-tuning datasets. The second dimension is model privacy. It concerns the safeguarding of internal model parameters, reasoning traces and outputs to prevent the recovery of embedded knowledge or original data.

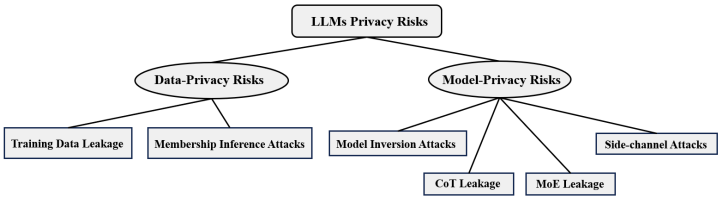

Figure 2. Classification of privacy risks in LLMs

Privacy risks exist from data collection to model deployment [7]. They appear in both the datasets used for learning and the computational mechanisms that process them. To analyze these risks clearly, this paper adopts a two-level analytical framework. Figure 2 shows this structure. The left branch represents data privacy risks, which focus on the exposure of sensitive data through Training Data Leakage and Membership Inference Attacks. The right branch represents model privacy risks, which include Model Inversion Attacks, CoT Leakage, MoE Leakage, and Side-channel Attacks. This classification gives a clear view of how data-level and model-level risks shape the privacy of LLMs.

3.1. Data privacy risks

3.1.1. Training data leakage

Training data leakage occurs when an LLM reproduces information from its training set [8]. The reproduced text may appear word for word, or it may rephrase the same meaning in slightly different words. The model can reveal names, identifiers, code, or private records that were never intended for public output. Such leakage affects both personal and corporate data. It can happen in pre-training when large, unfiltered datasets are used, or in fine-tuning when smaller but sensitive datasets are added. The mechanism comes from how LLMs learn. They predict the next token by fitting statistical patterns from massive corpora.

Training data leakage directly harms confidentiality. Private or proprietary information crosses the boundary between model training and public interaction. It may also harm integrity, since the model's output reflects individual records rather than general knowledge. A model that memorizes its data becomes less trustworthy and more prone to bias, turning its stored examples into unintended disclosures.

3.1.2. Membership inference attacks

Membership inference attack means that an adversary determines whether a specific record was used to train an LLM [6]. The attacker sends a query and observes the model's response. The goal is to judge if the sample exists in the training data. This form of attack targets privacy at the dataset level. It reveals participation information that should remain secret. The mechanism relies on statistical differences between trained and untrained samples. A model that has seen a sample usually gives higher confidence to that input. It produces lower loss and more stable predictions. A

DOI: 10.54254/2755-2721/2026.TJ31027

model that has not seen it produces less consistent probabilities. Attackers measure these differences. They use them to infer membership. Several attack strategies exist.

The impact on data confidentiality is serious. The disclosure of whether a user's record participated in training already violates privacy regulations. It can expose information about individuals, organizations, or datasets that were meant to remain anonymous. Model integrity also weakens, since confidence scores become channels for hidden data traces. A model that leaks membership cannot be fully trusted to separate training memory from general reasoning.

3.2. Model privacy risks

3.2.1. Model inversion attacks

Model inversion attack allows an adversary to reconstruct private data from an LLM [1]. The attacker observes model outputs or gradients and tries to rebuild the original input or its sensitive features. The goal is not only to guess class labels but to restore text, images, or personal attributes that the model has encoded. This attack exploits the correlation between model parameters and training data. The mechanism depends on how models represent patterns inside their parameters. During training, sensitive details become embedded in the weight space. Gradients and activations preserve traces of those details. When an attacker queries the model with partial inputs, the response distribution carries hidden signals. By analyzing those signals, the attacker estimates what content the model used during training. Several practical methods exist.

The attack harms data confidentiality because private information can cross the model boundary during inference. It also undermines model integrity. Once the attacker can map parameters back to original data, the model loses its role as a secure abstraction. A compromised model no longer separates learned knowledge from its sources. Its predictions become channels for unintended disclosure and may violate both user privacy and data governance laws.

3.2.2. CoT leakage

Chain-of-Thought (CoT) [9] leakage refers to the unintended disclosure of sensitive information through a model's intermediate reasoning steps [10]. Formally, let $f(x) = (r, y)$, where x is the user input, r is the sequence of intermediate reasoning tokens generated by the model, and y is the final answer. CoT leakage occurs when r contains sensitive attributes, identifiers, or verbatim fragments derived from training data or private user context, even when y itself is non-sensitive. The model produces intermediate steps to explain how it reaches an answer. These steps, known as chains of thought, may contain private content that was learned or inferred from training data. Leakage occurs because reasoning text is generated in natural language, which often carries unintended details about users, datasets, or hidden context. The mechanism comes from how LLMs handle multi-step reasoning. The model expands its answer into sequences of intermediate tokens. Each token reflects part of the internal state. If training data included personal records or specific examples, traces of those records may appear in the reasoning chain. Even when the final output looks safe, the intermediate steps may reveal sensitive fragments, identifiers, or relationships. Several forms of CoT leakage exist. The model may use actual data points as reasoning evidence. It may expose relationships between individuals or organizations that were embedded in the corpus. For example, a model may respond correctly to a benign arithmetic prompt yet reveal a memorized name or address within its explanatory steps. This phenomenon shows that the reasoning trace, rather than the final output alone, can expose private information encoded in the model.

DOI: 10.54254/2755-2721/2026.TJ31027

The effect on confidentiality is significant. Private data appears not in the final result but within the reasoning trace. Users and external systems can collect and analyze these traces. The effect on integrity is moderate but important. When reasoning relies on memorized examples, the logic no longer reflects general knowledge. The model becomes dependent on hidden data, reducing transparency and trust in its outputs.

3.2.3. MoE leakage

MoE leakage occurs when the routing and activation behavior inside a MoE model exposes information about data or internal processes [11]. In an MoE architecture, each input token is processed by a small subset of experts selected by a gating network. The routing decision depends on learned weights that reflect the characteristics of the input. These weights and routing patterns can leak details about the input data or reveal which experts store particular information. The mechanism comes from how the gating network transforms input representations into expert selections. Each token produces a distribution over experts. Sensitive features in the input affect this distribution. When attackers monitor routing statistics or activation scores, they can infer properties of the input that the model was designed to hide. In some cases, they can even identify which training data activated certain experts. MoE leakage can happen during inference or during distributed training. In inference, the routing log or performance trace may record which experts are used. If such logs are accessible, they expose correlations between tokens and data categories.

A concrete example of MoE leakage has been documented in the study “Stealing User Prompts from Mixture of Experts” by Yona et al [12].. This work demonstrates that MoE-based language models using Expert-Choice Routing (ECR) can unintentionally expose private user inputs through their routing mechanisms. The authors identify a specific vulnerability known as Tiebreak Leakage, in which adversaries exploit the deterministic resolution of routing conflicts among experts. In this setting, multiple users’ prompts are processed concurrently within a shared MoE inference environment. Because ECR assigns tokens to experts based on learned gating weights and deterministic tiebreak rules, small perturbations in an attacker’s query can alter the routing of co-batched tokens. By analyzing these routing changes and activation overlaps, the attacker can reconstruct the victim’s prompt content token by token, even without access to model outputs or gradients. Results show that this side effect arises from the intrinsic coupling between routing decisions and token identities. The leakage channel operates purely through the MoE layer’s expert-activation traces, revealing that sparsity and shared computation, while improving efficiency, introduce novel privacy exposure paths. This finding confirms that MoE leakage is not merely theoretical, but an empirically observed phenomenon that demands architectural and system-level mitigation.

MoE leakage harms confidentiality by revealing how data flows inside the model. Even if textual outputs look safe, routing traces can expose which tokens or topics are sensitive. It also affects integrity. When attackers exploit routing behavior, they can estimate the model’s internal structure and predict future outputs. This weakens the separation between computation and knowledge and threatens the trusted operation of the system.

3.2.4. Side-channel attacks

Side-channel attacks exploit runtime traces across the LLM stack [13]. These traces include timing, cache behavior [14], memory access, power use, and network traffic. Attackers collect these traces

DOI: 10.54254/2755-2721/2026.TJ31027

and infer internal states or input content [15]. They do not need model text outputs. They rely on physical and runtime side effects. See Figure 3 for the schematic and observation points.

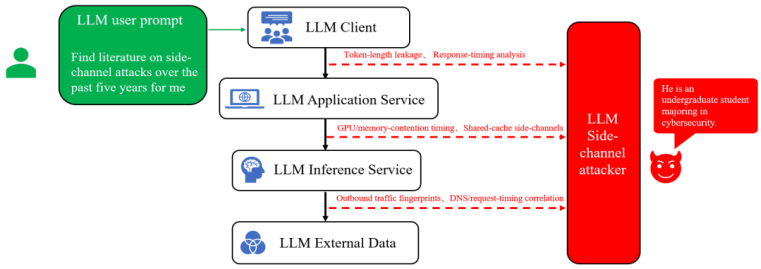

Figure 3. Side-channel attacks in LLMs

The figure shows four layers that produce side effects. The figure helps link observed traces to stack components. The top layer is the client and user interface. The client leaks token length and response timing. These signals reveal prompt structure and input length. Attackers use timing variance to infer partial prompts. The application layer exposes GPU contention and shared cache effects. Routing decisions change device load. Activation patterns alter cache usage. Attackers correlate contention and cache traces with token classes and data types. The inference layer generates outbound traffic and DNS timing signatures. Expert selection produces distinct packet sizes and network endpoints. Attackers map packet traces to expert routing and to topic categories.

Side channels break confidentiality. Attackers can infer private tokens and document types without reading outputs. They can infer user attributes from correlated traces. Side channels also weaken integrity. Attackers can craft queries that steer routing. Steering can force more revealing computation patterns.

Table 1. Side-channel signals and leakage paths in LLMs

Observation Layer	Signal Type	Main Source of Leakage	Possible Leaked Information
Client Layer	Timing, Token Length	Request / Response Cycle	Prompt structure, input pattern
Application Layer	GPU contention, Cache	Shared compute nodes and memory	Expert routing, token class distribution
Inference Layer	Network traffic	expert exchange and routing process	Topic category, expert activation map
External Data Layer	API calls, Service queries	databases and connected services	User identity, query history

Table 1 illustrates common side-channel signals in LLMs and their potential leakage paths across system layers. The table complements Figure 3 by mapping each signal to its layer of origin and possible exposure pattern.

4. Mitigation strategies

Effective mitigation of privacy risks in LLMs requires a layered defense framework. The mitigation landscape can be divided into three main technical levels and one regulatory dimension, as illustrated in Figure 4. The three technical levels are data-level, model-level, and system-level mitigation. Each level focuses on a different stage of the model lifecycle and addresses distinct types of privacy exposure. Data-level mitigation aims to control the sensitivity of the input and training data. It includes data cleaning, de-identification, and synthetic data generation. These methods

DOI: 10.54254/2755-2721/2026.TJ31027

reduce the presence of identifiable information before the model observes the data. Model-level mitigation focuses on the privacy of internal parameters and learned representations. It includes DP training, federated learning, CoT privacy control, and backdoor defense. These techniques restrict information leakage from inside the model. System-level mitigation targets the runtime environment and communication interfaces. It includes intrusion detection, cross-site attack protection, DoS and DDoS defense, and mitigation of man-in-the-middle attacks. These measures prevent external actors from intercepting or manipulating data during model operation.

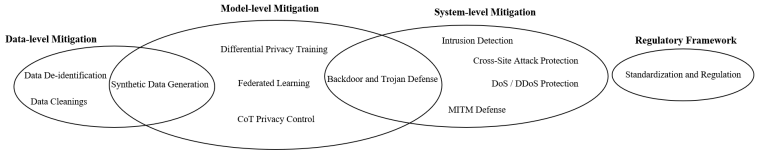

Figure 4. Classification of mitigation strategies in LLMs

4.1. Data-level mitigation

Data-level mitigation protects privacy at the earliest stage of model development. Its objective is to eliminate or transform sensitive information before it becomes part of the training set. This level of protection relies on three main approaches: data cleaning, de-identification, and synthetic data generation, which together form the foundation for secure and compliant model training.

• Data cleaning: Data cleaning acts as the first barrier against privacy leakage. It targets both explicit identifiers—such as names, phone numbers, and email addresses—and implicit signals that may reveal personal context. Modern pipelines combine rule-based filters, keyword dictionaries, and regular expressions with semantic analysis to identify these patterns automatically.

• Data de-identification: Data de-identification further minimizes privacy exposure by replacing identifiable elements with neutral or abstract substitutes. Substitution and hashing techniques mask personal attributes without altering data structure. In structured datasets, pseudonymization preserves internal correlations while concealing real identities. Statistical frameworks such as k-anonymity, l-diversity, and t-closeness provide quantitative guarantees that individual records cannot be singled out from a population.

• Synthetic data generation: Synthetic data generation offers a proactive solution when real datasets pose high privacy risks. By modeling statistical distributions and generating artificial samples, it replicates the structural characteristics of real data without disclosing individual details. Generative models like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) are commonly used to synthesize text, images, or structured records. In natural language processing, prompt-controlled generation can produce paraphrased sentences or context-preserving rewrites that mimic language style while masking personal content.

4.2. Model-level mitigation

Model-level mitigation protects privacy and integrity within the model itself. While data-level defenses address what enters the model, model-level strategies ensure that its internal mechanisms, parameters, and reasoning processes do not reveal or distort sensitive information.

• CoT privacy control: CoT privacy control addresses the problem of reasoning leakage. During step-by-step reasoning, large models may inadvertently reproduce fragments of training data or sensitive intermediate details. To prevent this, CoT filtering restricts the exposure of internal reasoning traces in both training and inference stages. Selective decoding hides intermediate

DOI: 10.54254/2755-2721/2026.TJ31027

reasoning steps when they are not required for output interpretability [16]. Regularization terms can further penalize memorization tendencies, encouraging generalized reasoning rather than verbatim recall.

• DP training: Differential Privacy (DP) [17] training provides statistical guarantees that individual samples in the training set cannot be inferred from the trained model [18]. In DP-based learning, calibrated noise is added to gradients during optimization, limiting the contribution of any single record. The strength of privacy protection is determined by the privacy budget ε, which controls the trade-off between data utility and confidentiality. Implementations such as DP-SGD integrate seamlessly with large-scale training pipelines and are now widely adopted in privacy-critical applications.

• Federated learning: Federated learning extends this principle by removing the need for centralized data aggregation [19]. In a federated system, multiple clients train local models on their private data and share only gradient updates with a central server. The server aggregates these updates into a global model without ever accessing raw data. This structure prevents direct exposure of private content and enables collaboration among institutions that cannot share datasets directly, such as hospitals or financial organizations. However, federated training introduces new challenges, including communication efficiency, client drift, and the potential leakage of gradient information.

• Backdoor and Trojan defense: Backdoor and Trojan defense ensures the trustworthiness of model parameters during and after training. A backdoor attack embeds hidden triggers that cause malicious behavior when specific inputs are presented. Trojanized models may output targeted responses or leak encoded data once activated. At the model layer, defense focuses on identifying and removing such manipulations before deployment. Techniques include neuron pruning, gradient-based anomaly detection, and fine-tuning sanitization, which remove suspicious activation patterns or biased weights.

• Synthetic data: Synthetic data also plays a role in protecting privacy at the model level. Unlike data-layer synthesis, which replaces real samples before training, model-level synthetic data is used to simulate sensitive domains or augment underrepresented patterns without accessing actual user data. Synthetic fine-tuning datasets can teach models new capabilities while ensuring that private corpora remain untouched.

4.3. System-level mitigation

System-level mitigation strengthens the infrastructure that supports model deployment and interaction. While data-level and model-level defenses focus on internal privacy, system-level measures prevent information leakage, unauthorized access, and service disruption in real-world environments. Key components include Man-in-the-Middle (MitM) defense, cross-site attack protection, DoS/DDoS protection, intrusion detection, and backdoor and trojan defense.

• MitM defense: Man-in-the-Middle (MitM) defense protects the communication channels that connect clients, application services, inference services, and external data resources [20]. As shown in Figure 5, an attacker may intercept interactions at multiple stages, including client-side inputs, plugin connections, inference routing, and retrieval processes. Each interception point represents a potential risk of data manipulation or information theft. At the client level, malicious browser extensions or injected scripts can capture prompts and responses. Within the application service, attackers may tamper with plugin connections or poison prompts transmitted to the inference engine. In the inference layer, modified requests or injected vectors can alter model outputs. Even external data sources such as vector databases may be compromised through poisoned retrieval entries.

DOI: 10.54254/2755-2721/2026.TJ31027

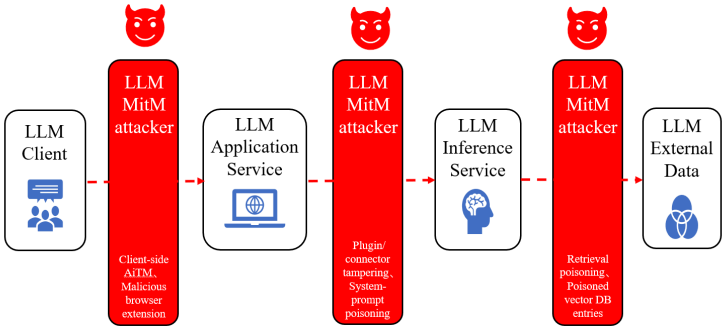

Figure 5. MitM attack paths across LLM layers

• Cross-site attack protection: Cross-site attack protection safeguards LLM-based web interfaces and API endpoints from malicious script injection or data theft. Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) exploit domain trust to inject unauthorized commands or extract private data. Defense mechanisms include strict input sanitization, token-based authentication, and secure cookie settings with appropriate SameSite attributes. Content Security Policy (CSP) headers and origin verification prevent untrusted scripts from executing within web-integrated model environments.

• DoS and DDoS protection: DoS and DDoS protection ensures the availability and stability of LLM services under malicious traffic. Because of their high computational demands, LLMs are particularly vulnerable to resource exhaustion. Rate limiting and adaptive throttling manage excessive requests from individual clients, while distributed filtering systems mitigate large-scale attacks at the network edge. Caching frequently used responses and isolating inference queues maintain responsiveness during high loads, and redundant routing with auto-scaling infrastructure sustains continuous operation.

• Intrusion detection: Intrusion detection provides continuous surveillance over runtime environments to identify abnormal behavior and potential breaches. Host-based and network-based systems monitor process activity, system calls, and network flows to detect intrusions in real time. Machine learning models trained on system logs can recognize deviations that indicate privilege escalation or compromised components. Continuous auditing of containerized deployments ensures that model weights, configurations, and dependencies remain intact.

• Backdoor and Trojan defense: Backdoor and Trojan defense operates at the intersection of model and system security. Backdoors represent hidden manipulations within model weights or pipelines that trigger malicious behavior under specific inputs. At the system level, such threats may arise from compromised deployment scripts, malicious updates, or infected hardware accelerators. Static and dynamic scanning tools inspect binaries, container images, and hardware interfaces to locate injected code or concealed triggers. Runtime attestation and Trusted Platform Modules (TPMs) verify the integrity of execution environments through hardware-backed trust.

4.4. Standardization and regulation

Standardization and regulation form the governance foundation of privacy protection. In China, the Data Security Law [21] and the Personal Information Protection Law [22] establish explicit requirements for lawful data collection, storage, and cross-border transmission. At the international level, frameworks such as the General Data Protection Regulation [23] of the European Union and the OECD Privacy Guidelines provide the global foundation for data protection and accountability.

DOI: 10.54254/2755-2721/2026.TJ31027

To enhance the practical usability of governance norms, this subsection also provides compliance operation checklists for enterprise-internal LLM deployments and public API services.

For internal enterprise deployments of LLMs, organizations can develop end-to-end governance pipelines that enforce data-minimization requirements, integrate mandatory data-classification processes before model ingestion, and maintain strict authorization controls supported by continuous audit logging. In parallel, enterprises can maintain a formal incident-response workflow capable of rapidly containing privacy breaches and performing traceable forensic analysis.

Public API service providers can anchor compliance in transparent operational disclosures that explain data retention periods, logging boundaries, and any conditions under which cross-border transfers occur. Privacy protection can rely on strong workload isolation between tenants, cryptographically enforced separation of execution contexts, and configurable privacy modes that disable persistent storage or model retraining from user data. These integrated governance pathways convert high-level regulatory expectations into actionable procedures, enabling predictable and verifiable privacy guarantees across heterogeneous LLM deployment settings.

4.5. User-side privacy protection mechanisms

Effective privacy protection in LLM deployments should also extend to the user side. To strengthen the full-lifecycle protection framework, user-side defense mechanisms can be introduced at the interaction and interface levels. Prompt desensitization tools automatically identify and remove sensitive elements such as personal identifiers, account numbers, or confidential business terms before the user input is transmitted to the model. These tools may combine pattern-based filtering, named-entity recognition, and customizable privacy dictionaries to achieve an appropriate balance between usability and confidentiality. In addition, privacy-friendly interaction design can reduce information disclosure during user–model communication. Typical approaches include default ephemeral storage of conversation history, on-device pre-processing of textual inputs, and transparent interfaces that clearly indicate what information is stored or transmitted. Integrating these user-oriented safeguards with the institutional and technical measures presented in previous sections enables organizations to establish an end-to-end privacy assurance framework that covers governance, infrastructure, and direct user engagement.

4.6. Scale-specific privacy risks and adaptive protection strategies

The privacy exposure of LLMs also differs across model scales. This section conducts a comparative analysis of how privacy risks evolve with parameter size and examines adaptive protection strategies suited to small-scale, medium-scale, and large-scale LLMs. The goal is to clarify the relationship between scale, risk patterns, and feasible mitigation measures.

• Small-Scale LLMs (0–10B parameters): Limited contextual comprehension and simplified architectures make small models more likely to capture or reproduce explicit sensitive information. They are also vulnerable to basic extraction attacks due to weak structural defenses. To mitigate these risks, input desensitization tools can automatically filter sensitive content before model processing. Enforcing data minimization principles and using only public, non-sensitive datasets further reduce privacy exposure. Knowledge distillation from larger, privacy-hardened models can enhance risk awareness without increasing model size.

• Medium-Scale LLMs (10B–100B parameters): As reasoning capability improves, medium-scale models face inferential privacy risks, where they deduce hidden personal trait from non-sensitive text. They also become more susceptible to membership inference attacks that reveal whether

DOI: 10.54254/2755-2721/2026.TJ31027

specific samples were used in training. Effective defense requires introducing differential privacy during fine-tuning with a calibrated privacy budget, and employing federated learning to prevent data centralization. Regular inferential-privacy audits using synthetic data can further detect and block sensitive inference paths before deployment.

• Large-Scale LLMs (100B+ parameters): At larger scales, extended contextual memory and generalization capabilities amplify both memorization-based leakage and prompt manipulation risks. Models may reproduce rare or proprietary data or respond to adversarial prompts that elicit hidden content. Multi-layer defenses are essential: adaptive-noise DP-SGD during training to suppress memorization, prompt-engineering guardrails to block malicious injections, and model watermarking for traceability of outputs. Strong access control, encrypted computation for privacy-critical inference, and pre-deployment red teaming further strengthen system resilience against large-scale compromise.

5. Quantitative comparative evaluation framework

To complement the qualitative analysis presented above, future research can incorporate quantitative comparative experiments to evaluate the effectiveness and efficiency of different privacy protection strategies for LLMs. Such experiments would provide measurable evidence for balancing privacy, utility, and performance trade-offs across data-level, model-level, and system-level defenses.

Quantitative evaluations can be conducted along multiple analytical dimensions, including but not limited to:

• Time Complexity: computational overhead introduced by privacy-preserving mechanisms.
• Space Complexity: additional memory or storage consumption.
• Latency: response delay under privacy-enhanced inference settings.
• Privacy Risk Probability: statistical likelihood of data leakage or membership inference.
• Concurrency or Throughput: model performance under simultaneous user interactions.

These metrics establish a unified framework for comparing privacy strategies across heterogeneous LLM architectures. Future work will focus on implementing controlled experiments under these quantitative dimensions to provide empirical validation of the proposed mitigation framework.

6. Conclusion

This paper has presented a comprehensive review of data privacy risks in LLMs. It analyzed the mechanisms through which private information may be exposed and classified the risks into data-level, model-level, and system-level dimensions. The review showed that LLMs face a wide spectrum of vulnerabilities, including training data leakage, inference-based reconstruction, and side-channel observation. It also examined mitigation strategies that address these problems from multiple layers. Data-level measures focus on cleaning, de-identification, and synthetic data generation. Model-level protection includes DP, federated learning, and backdoor defense. System-level strategies involve encryption, intrusion detection, and trusted hardware.

References

[1] Li, Y., Tramèr, F., & Carlini, N. “Privacy Risks of Large Language Models: A Survey.” arXiv: 2401.14403, 2024.
[2] Fedus, W., Zoph, B., & Shazeer, N. “Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity.” arXiv: 2101.03961, 2021.
[3] Zeng, S., et al. “Exploring Memorization in Fine-tuned Language Models.” ACL 2024, 2024.

DOI: 10.54254/2755-2721/2026.TJ31027

[4] Cyberspace Administration of China. Interim Measures for the Management of Generative Artificial Intelligence Services, 2023.
[5] Standardization Administration of China. GB/T 45652-2025 Information Security Technology — Security Requirements for Training Data of Generative AI Models, 2025.
[6] Lukas, N., Struppek, L., & Kersting, K. “Analyzing Leakage of Personally Identifiable Information in Language Models.” arXiv: 2302.00539, 2023.
[7] Chen, K., et al. “A Survey on Privacy Risks and Protection in Large Language Models.” arXiv: 2505.01976, 2025.
[8] Carlini, N., et al. “Extracting Training Data from Large Language Models.” USENIX Security, 2021.
[9] Wei, J., et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.” NeurIPS, 2022.
[10] Green, T., et al. “Leaky Thoughts: Large Reasoning Models Are Not Private Thinkers.” arXiv: 2506.15674, 2025.
[11] Lai, Z., et al. “SAFEx: Analyzing Vulnerabilities of MoE-Based LLMs via Safety Control Experts.” arXiv: 2506.17368, 2025.
[12] Yona, G., et al. (2024). “Stealing User Prompts from Mixture of Experts.” arXiv: 2410.22884.
[13] Song, L., et al. “Unveiling Timing Side Channels in LLM Serving Systems.” arXiv: 2409.20002, 2024.
[14] Gao, Z., et al. “Unveiling Hardware Cache Side-Channels in Local Large Language Models.” USENIX Security 2025, 2025.
[15] Zheng, X., et al. “Inputsnatch: Stealing Input in LLM Services via Timing Side-Channel Attacks.” arXiv: 2411.18191, 2024.
[16] Diao, S., et al. “Active Prompting with Chain-of-Thought for Large Language Models.” arXiv: 2302.12246, 2023.
[17] Dwork, C., et al. “Calibrating Noise to Sensitivity in Private Data Analysis.” TCC, 2006.
[18] Abadi, M., et al. “Deep Learning with Differential Privacy.” ACM CCS, 2016.
[19] Kairouz, P., et al. “Advances and Open Problems in Federated Learning.” Foundations and Trends in ML, 2021.
[20] Das, B. C., Amini, M. H., & Wu, Y. “Security and Privacy Challenges of Large Language Models: A Survey.” arXiv: 2402.00888, 2024.
[21] Standing Committee of the National People’s Congress. Data Security Law of the People’s Republic of China, 2021.
[22] Standing Committee of the National People’s Congress. Personal Information Protection Law of the People’s Republic of China, 2021.
[23] European Union. General Data Protection Regulation (EU) 2016/679, 2016.